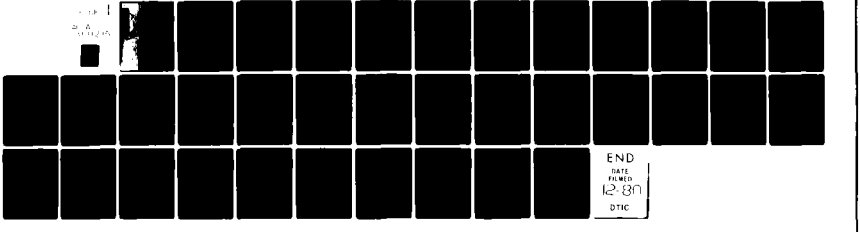
AD-A091 215 TEXAS UNIV AT AUSTIN CENTER FOR CYBERNETIC STUDIES F/G 15/5
RECENT DEVELOPMENTS IN COMPUTER IMPLEMENTATION TECHNOLOGY FOR N--ETC(U)
JUL 80 F GLOVER, D KLINGMAN N00014-80-C-0242
UNCLASSIFIED CCS-377 NL



END
DATE FILMED
12-80
DTIC

LEVEL II (12)

CENTER FOR CYBERNETIC STUDIES

The University of Texas
Austin, Texas 78712

CCS 377

RECENT DEVELOPMENTS IN COMPUTER IMPLEMENTATION TECHNOLOGY FOR NETWORK FLOW ALGORITHMS

by

Fred Glover*

Darwin Klingman**

July 1980

* Professor of Management Science, Graduate School of Business, University of Colorado, Boulder, CO 80309

** Professor of Operations Research and Computer Sciences, Department of General Business, BEB 608, The University of Texas at Austin, Austin, TX 78712

This research was partially supported by Office of Naval Research Contracts N00014-78-C-0222 and N00014-80-C-0242 with the Center for Cybernetic Studies, The University of Texas at Austin, and by Department of Transportation Contract DOT-RC-92039. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

CENTER FOR CYBERNETIC STUDIES

A. Charnes, Director
BEB 203E
The University of Texas at Austin
Austin, Texas 78712
(512) 471-1821

ABSTRACT

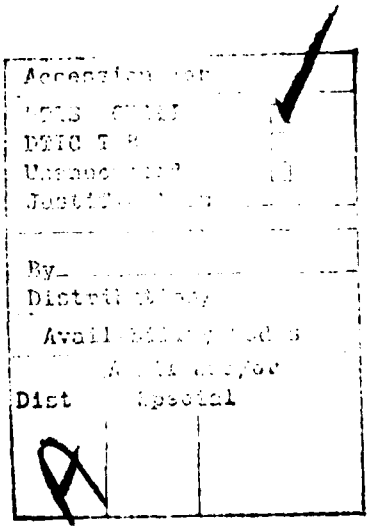
The application of computer implementation technology to network optimization has brought about unprecedented advances in solution efficiency. The remarkable gains of the early to mid 1970's for solving transportation and transhipment problems are widely known, enabling network codes to out-perform LP codes by two orders of magnitude for these problems. The pioneering study by Gilsinn and Witzgall demonstrated that effective use of computer implementation technology could reduce solution times for shortest path problems from one minute to slightly more than one second, using the same general shortest path algorithm, computer, and compiler.

The momentum launched by these studies has not dwindled, but continues into the present. New advances in all areas of network optimization have recently superseded the procedures previously found to be best. Latest computer implementations clearly outstrip the best codes of the recent past as our understanding of the important relation between algorithmic design and implementation continues to grow.

We undertake to report on some of the major computer implementation studies of the past few years and to present preliminary results on the new developments.

KEY WORDS

Shortest Path
Assignment Problems
Transportation Problems
Network Flow
Transhipment Problems



INTRODUCTION

The historical development of network algorithms has paralleled and contributed to the growth of an important interface between mathematics and computer science, called *computer implementation technology*. This technology, which started back in 1952 with the implementation of the Stepping Stone Method on the National Bureau of Standards Eastern Automatic Computer, has been recognized only recently as a major discipline in its own right.

Computer implementation technology seeks to determine efficient special procedures for carrying out subalgorithms of a general method on a digital computer by investigating (1) what kinds of information to generate and maintain for executing operations most effectively, (2) which data structures are best to record, access and update this information, and (3) what methods are most suitable for processing these data structures to make the desired information available when it is needed.

To garner such knowledge requires experimentation that artfully blends the best elements of mathematics and computer science. The computer is used both to evaluate the efficiency of resulting algorithmic processes (embodied in executable programs) and to provide statistics about the operation of key components under varying test conditions. Properly generated and utilized, these statistics allow researchers to gain valuable insights on how to improve the design of these components. This iterative modification, integration, and evaluation of key processes is directly analogous to the laboratory

research of other disciplines and leads us to view computer implementation technology as the laboratory research of mathematics and computer science.

The application of computer implementation technology to network optimization has brought about unprecedented advances in solution efficiency. The remarkable gains of the early to mid 1970's for solving transportation and transshipment problems are widely known, enabling network codes to out-perform LP codes by two orders of magnitude for these problems. The pioneering study by Gilsinn and Witzgall demonstrated that effective use of computer implementation technology could reduce solution times for shortest path problems from one minute to slightly more than one second, using the same general shortest path algorithm, computer, and compiler.

The momentum launched by these studies has not dwindled, but continues into the present. New advances in all areas of network optimization have recently superseded the procedures previously found to be best. Latest computer implementations clearly outstrip the best codes of the recent past as our understanding of the important relation between algorithmic design and implementation continues to grow.

We undertake to report on some of the major computer implementation studies of the past few years and to present preliminary results on the new developments.

SHORTEST PATH ALGORITHMS

Shortest path analysis (finding the shortest paths from a single node to all other nodes in a directed network unless otherwise specified)

was one of the first network areas to benefit greatly from the application of computer implementation technology [35].

The importance of shortest path solutions in quantitative transportation and communication models has caused a great deal of activity in this area, generating a proliferation of shortest path algorithms in the literature. However, more careful consideration discloses there are only a handful of *general* shortest path methods, each containing a number of *subalgorithms*. These latter are designed to handle special subproblems or sets of operations such as finding the minimum of a set, breaking a loop, reconnecting subtrees, carrying out computations over the nodes and arcs of subtrees, etc.

The many different ways developed to handle these subproblems, unfortunately, have often been referenced as different algorithms rather than as variants of the small class of general algorithms. Contributions toward a unified framework for shortest path algorithms have been made by Denardo and Fox [17], Dial, Glover, Karney, and Klingman [19], Dreyfus [23], and Gilsinn and Witzgall [35].

Gilsinn and Witzgall [35] were the first to conduct an extensive computational study. They found that the Dijkstra algorithm [20], using an address calculation sort proposed by Dial [18], was the most efficient of the methods tested. The subsequent study of [19] evaluated an expanded range of solution procedures and problem structures. This study showed the most efficient method depended on problem topology. For relatively dense random problems, the best implementation of Dijkstra's procedure was dominated by an implementation of Dantzig's algorithm [15]

(using an address calculation sort in both cases). For sparse random and grid problems the best Dantzig implementation was in turn inferior to a label-correcting procedure due to Pape [56] (designed to process a candidate node list from both ends). A by-product of this work was to demonstrate the perhaps surprising result that all of the tested shortest path methods could be interpreted as special variants of the primal simplex method. (Computer listings of the best codes, all in FORTRAN, can be obtained from the authors.)

Sequels to the study of [19] have been undertaken by Klingman, Mote, and Whitman [46], Elam, Klingman, and Mulvey [25], and Klingman and Mulvey [47], extending the best algorithms of [19] to "in-core out-of-core" implementations and mini-computers. The study of Denardo and Fox [17] has further developed new "multiple bucket" algorithms that are projected to be efficient for extremely large problems. The problems tested in their study, however, were solved more efficiently by essentially the same label-correcting procedure found best for sparse random problems in [19]. (Extrapolation of solution times for the multiple bucket algorithms suggest the possibility that the multiple bucket methods may dominate for some larger problem size.)

Another study by Florian, Nguyen, and Pallottino [28] focuses on advanced dual methods for problems of finding shortest paths between all pairs of nodes (in contrast to finding shortest paths from a single source to all others), and reports promising gains in efficiency for these problems. This work provides a method for the "all node pairs" (multiple-origin) problem that is 37% faster than an implementation of

Pape's method [56] applied in iterative fashion over successive origins. It is important to note, however, that the first shortest path problem is solved using an implementation of Pape's method.

The latest advance, still in a developmental stage, is a new label-correcting code by the authors employing a threshold/partitioning scheme for candidate node management. This procedure appears to give it many of the advantages of a label-setting code, yet with greater efficiency than the previously best codes of either the label-setting or label-correcting type.

Preliminary testing on 40 sparse random problems containing 10,000 arcs and 1250 to 2000 nodes yields solution times 36% to 44% faster than the best code, called Pape [56], for these structures.

Table I contains the mean solution times over ten problems for each problem size. Also Table I indicates the number of nodes scanned by the new code, called THRESH. The results show that the number of nodes scanned is only slightly larger than the total number of nodes, which gives an indication of an upper limit on its efficiency. If these results carry over to other problem topologies, they will not only establish the superiority of this new method for single-origin shortest path problems, but also for "all node pairs" shortest path problems approach in [28].

ASSIGNMENT PROBLEMS

Assignment problems have had a colorful history. Representing an important special case of the transportation problem, they were among

TABLE I

MEAN SOLUTION TIMES (CPU SECONDS ON A CDC 6400 USING THE FORTRAN FTN COMPILER) FOR SHORTEST PATH PROBLEMS WITH 10,000 ARCS; DISTANCES 1-200. (10 Problems were solved for each node size)

No. of Nodes	2000	1667	1429	1250
PAPE	4.2	4.7	4.9	4.7
THRESH	2.7	2.8	2.7	2.9
THRESH No. of Nodes Scanned	2150	1747	1511	1371

the first network problem classes to be considered worthy of investigation in their own right, and a number of different algorithms were proposed for solving them. For a number of years Kuhn's "Hungarian method" [50] was judged to be best, though a specialization of the out-of-kilter method later appeared to provide the best computer times [41]. The first departure from these "classical" approaches occurred with the development of the alternating path basis (AP-AB) primal simplex method for networks [5, 6], which turned out to yield a highly effective specialization for assignment problems. The AP-AB procedure implemented by Barr, Glover, and Klingman [5] appeared to be somewhat faster than the best previous code of [41], and to utilize less memory.

Subsequently, a great deal of new activity has taken place in the assignment area. Hung and Rom [43] developed a "recursive" method for exploiting the AP-AB structure that is twice as fast as the AP-AB code of [5] for totally dense problems. Since the Hung and Rom code was developed for totally dense problems, it is substantially slower than the AP-AB primal simplex code [5] on non-dense problems. (See Table II.)

At about the same time, Weintraub [60] tested an implementation of the Edmonds and Karp assignment procedure that proved still more efficient than the Hung and Rom code. (See Table II.) G. L. Thompson [59] developed a special assignment procedure of yet another type that appears highly promising, but that has not been tested against alternative methods.

TABLE II

SOLUTION TIMES IN SECONDS ON A DUAL CYBER 170/175
USING THE FTN COMPILER FOR 200 x 200 ASSIGNMENT
PROBLEMS WITH COST RANGE 1-100

	NUMBER OF ARCS				
CODE	1500	2250	3000	3750	4500
AP-AB (Barr, et al)	.483	.603	.634	.684	.916
RELAX (Rom-Hung)	1.365	1.462	1.168	1.054	1.157
Dual (Weintraub)	.173	did not achieve optimality	.272	.291	.343
SPAN (Engquist)	.080	.175	.150	.282	.182
PDQIK (Glover, Glover and Klingman)	.065	.092	.107	.091	.127

* The times reported in this table are the results of testing conducted by Professor Michael Engquist and not by the authors. We greatly appreciate being permitted to use them.

More recently, a comparative study of assignment methods has been undertaken by Engquist [27]. Engquist has observed that the assignment procedure due to Dinic and Kronrod [22] is similar to the one developed by Hung and Rom [43], though derived from entirely different conceptual bases. The study of Engquist shows that a refined implementation of Dinic and Kronrod's procedure, which is embodied in a code called SPAN, is very efficient. (See Table II.)

Another innovation in assignment solution procedures, which was developed after Hung and Rom's work but before Engquist's work, was motivated by the advance in shortest path methods embodied in the THRESH code. The thresholding and partitioning strategies of THRESH have been extended by F. Glover, R. Glover, and Klingman to accelerate successive optimization steps of a "primal-dual" type of algorithm. Building on refinements in primal-dual methodology due to Ellis Johnson [44] and adding further refinements to capitalize on the THRESH strategies, the resulting assignment procedure, called PDQIK, appears substantially superior even to Engquist's streamlined version of Dinic and Kronrod's method. (See Table II.)

The assignment problems of Table II were generated using NETGEN [49] and the problem specification given in [49] for problems numbered 11-15. (See Table VIII.)

MAXIMUM FLOW PROBLEMS

For a number of years the maximum flow network problem has attracted the attention of prominent researchers in network optimization. Since

the ground-breaking work of Ford and Fulkerson, a variety of algorithms featuring good "worst-case" bounds have been proposed for this problem. Surprisingly though, there have been almost no empirical evaluations of these algorithms.

Cheung [11] recently conducted the first significant computational investigation of maximum flow methods, testing several of the major approaches. Although an important step in the right direction, Cheung's implementations employ methodology and data structures originating at least a dozen years ago. That is, the many advances in network implementation technology of the past decade were not incorporated into these codes.

To remedy this situation, Glover, Klingman, Mote, and Whitman [39] tested maximum flow implementations designed to make effective use of the recent developments in network labeling and data organization techniques. To safeguard against being swayed too heavily by preliminary analyses (and past experience in other network settings), the study implemented more than one type of data structure and associated processing techniques for the algorithms tested. Additionally, the resulting codes were tested on four distinct problem topologies.

The investigation included in its examination the two most widely heralded general classes of algorithms for maximum flow network problems--the label tree and referent algorithms. Over 50 codes were developed and at least partially tested for these methods. In the process, a new member of the referent class of algorithms was developed, called the *sub-referent method*, which proved far more effective than all others.

In addition, the study investigated a third type of approach which constitutes a special-purpose variant of the primal simplex method. Previously, researchers had neglected primal methods in favor of more classical labeling types of algorithms, primarily because simple choice rules yield good worst-case bounds for these methods. Recently, Cunningham [12, 13] has partly removed the theoretical bias against primal simplex maximum flow methods by deriving a computational bound for one of its variants (different from the one tested in [39]). Although this theoretical bound is not nearly as good as those for label tree and referent algorithms, practical experience in the network area over the past decade argues strongly for testing a derivative of the primal simplex method, which has proved highly robust and effective in other settings.

The study of [39] tested over twenty implementation variants of the primal simplex method. Independently, Grigoriadis and Hsu [40] recognized the lack of testing in the literature of the primal simplex method on maximum flow problems. Accordingly, they applied a modified version of their code, RNET, designed for solving minimum cost flow problems, and found it surprisingly efficient [40].

Two of the approximately 70 algorithmic implementations of label tree, referent and primal methods tested in [39] stand out far above all the rest. One of these is an implementation of the new sub-referent method which is dramatically faster than all contenders on two of the problem topologies tested. The other method that stands above the rest

of the field is an implementation of a variant of the primal simplex method, called SEQCS, which is fastest on two of the problem topologies and is second in efficiency to the sub-referent method on the other two problem topologies. In addition, the primal simplex variant requires approximately one-third the computer memory required by other algorithms, and lends itself more readily to an efficient in-core out-of-core implementation. A rather surprising result of this study [39] is that even the best label tree codes, which incorporate the "collective wisdom" of many contributors [11, 16, 24, 32, 34] to the original label tree algorithm of Ford and Fulkerson [30], do not compete favorably against either referent or simplex based codes.

The development and testing of Grigoriadis and Hsu [40] of a modified version of RNET on maximum flow problems raises the intriguing question of whether--or to what extent--it is useful to develop fully specialized primal simplex methods for maximum flow problems. Accordingly, we undertook to investigate this issue by testing both specialized and semi-specialized primal simplex codes on the same maximum flow problems, computer, and compiler.

The specialized primal simplex code used in our test is the SEQCS code described in [39] and the semi-specialized primal simplex code is RNET [40] using the maximum flow problem option.* The tests were conducted on four types of graph topologies: random (R), multi-terminal (MR), transit grid (TG), and hard (H). Tables III, IV, V, and VI con-

---

* We slightly modified RNET to correct an error in its pricing routine for maximum flow problems.

tain the specifications of each topology type, respectively. A description of how these problems were generated is contained in [39].

TABLE III

RANDOM PROBLEM SPECIFICATIONS

PROBLEM	|N|	|A|	ARC CAPACITY RANGE
R1	250	1250	1-100
R2	250	1875	1-100
R3	250	2500	1-100
R4	500	2500	1-100
R5	500	3750	1-100
R6	500	5000	1-100
R7	750	3750	1-100
R8	750	5825	1-100
R9	750	7500	1-100
R10	1000	5000	1-100
R11	1000	7500	1-100
R12	1000	10000	1-100

TABLE IV

MULTI-TERMINAL RANDOM PROBLEM SPECIFICATIONS

PROBLEM	|N|*	|A|	AVERAGE NO. OF ARCS INCIDENT ON EACH MASTER SOURCE (TERMINAL)	ARC CAPACITY RANGE**
MR1	250	1250	5.0	1-100
MR2	250	1875	7.5	1-100
MR3	250	2500	10.0	1-100
MR4	500	2500	5.0	1-100
MR5	500	3750	7.5	1-100
MR6	500	5000	10.0	1-100
MR7	750	3750	5.0	1-100
MR8	750	5825	7.5	1-100
MR9	750	7500	10.0	1-100
MR10	1000	5000	5.0	1-100
MR11	1000	7500	7.5	1-100
MR12	1000	10000	10.0	1-100

*There were five master source nodes and five master terminal nodes.

**Excluding arcs entering or leaving source and terminal nodes.

TABLE V

TRANSIT GRID PROBLEM SPECIFICATIONS

PROBLEM	|N|*	|A|	AVERAGE NO. OF ARCS INCIDENT TO EACH MASTER SOURCE (TERMINAL)	ARC CAPACITY RANGE**
TG1	235	1240	40	1-100
TG2	235	1640	80	1-100
TG3	410	2120	60	1-100
TG4	410	2720	120	1-100
TG5	635	3200	80	1-100
TG6	635	4000	160	1-100
TG7	910	4480	100	1-100
TG8	910	5480	200	1-100

*Including five master source nodes and five master terminal nodes.

**Excluding arcs entering or leaving master source and master terminal nodes.

TABLE VI

HARD PROBLEM SPECIFICATIONS

PROBLEM	|N|	|A|	ARC CAPACITY RANGE
H1	20	190	1-82
H2	40	780	1-362
H3	60	1770	1-782
H4	80	3160	1-1522
H5	100	4950	1-2402

All computer runs were carried out on the Dual CYBER 170/175 using the MNF FORTRAN compiler during periods of comparable computer use. A total of 185 problems were solved. The results, reported in Table VII, provide median solution times for a group of five problems (i.e., five different problems of the same dimension were generated and solved). RNET was run using five different pivot strategies. The pivot strategies tested varied in the arc pricing frequency used for each pass through the arc list. This is controlled by the user supplied parameter, FRQ, in RNET. The heading in Table VII indicates the value used for this parameter (e.g., RNET 1 means FRQ = 1).

TABLE VII

COMPUTER TIMES* IN SECONDS FOR MAXIMUM FLOW PROBLEMS
ON A DUAL CYBER 170/175 USING MNF COMPILER

PROBLEMS	SEQCS	RNET 20	RNET 10	RNET 5	RNET 2	RNET 1
R1	.04	.12	.12	.12	.18	.26
R2	.07	.21	.16	.17	.22	.31
R3	.08	.16	.17	.18	.22	.31
R4	.07	.24	.25	.27	.38	.54
R5	.16	.35	.35	.34	.43	.61
R6	.24	.40	.35	.35	.48	.69
R7	.12	.38	.35	.42	.55	.81
R8	.22	.44	.45	.46	.66	.89
R9	.40	.67	.57	.60	.71	.98
R10	.21	.50	.54	.53	.77	1.08
R11	.32	.65	.66	.67	.89	1.18
R12	.46	.83	.78	.84	.99	1.34
TOTAL	2.39	4.95	4.75	4.95	6.48	9.00
MR1	.11	.24	.20	.19	.26	.36
MR2	.25	.50	.41	.39	.43	.55
MR3	.25	.36	.36	.35	.39	.49
MR4	.13	.33	.31	.35	.39	.59
MR5	.38	.51	.63	.44	.58	.76
MR6	.67	1.09	.81	.72	.73	1.00
MR7	.30	.53	.56	.46	.67	.90
MR8	.45	.67	.57	.57	.77	1.07
MR9	1.11	2.45	1.50	1.33	1.22	1.52
MR10	.32	.82	.62	.66	.89	1.24
MR11	.86	1.16	1.06	1.12	1.22	1.49
MR12	1.67	2.14	1.89	1.64	1.43	1.73
TOTAL	6.50	10.80	8.92	8.22	7.76	11.70
H1	.03	.06	.06	.08	.11	.17
H2	.20	.43	.45	.51	.59	.87
H3	.66	1.50	1.51	1.37	1.88	2.50
H4	1.53	3.97	3.44	3.23	3.96	4.85
H5	2.96	7.58	6.54	6.12	7.61	9.27
TOTAL	5.38	13.54	12.00	11.31	14.15	17.66
TG1	.09	.21	.19	.24	.30	.44
TG2	.08	.20	.18	.23	.33	.46
TG3	.21	.39	.39	.42	.56	.85
TG4	.18	.35	.39	.40	.56	.85
TG5	.35	.64	.64	.67	.89	1.31
TG6	.27	.57	.58	.61	.82	1.24
TG7	.43	.89	.88	.93	1.14	1.70
TG8	.51	.96	.89	.99	1.25	1.93
TOTAL	2.12	4.21	4.14	4.49	5.85	8.78
GRAND TOTAL	16.39	33.50	29.81	28.97	34.24	47.14

*Five problems of each type were solved and the solution time reported.

The times in Table VII indicate that SEQCS strictly dominates RNET on all problem topologies. The degree of dominance varies substantially, however. For instance, SEQCS is approximately twice as fast as the best RNET times, RNET 10, on the grid problems, but it is only 20% faster than RNET 2 on the multi-terminal random problems.

Overall SEQCS is approximately 80% faster than the best RNET, RNET 5. Since SEQCS requires less computer memory and its times are notably better, it appears worthwhile developing totally specialized primal simplex codes for applications requiring repeated solution of maximum flow problems.

MINIMUM COST FLOW PROBLEMS

All of the problems previously discussed may be viewed as a subclass of the minimum cost flow (MCF) problem class. Since the inception of mathematical programming, this problem class has received considerable attention not only from MCF practitioners and theoreticians, but from practitioners and theoreticians across all fields of mathematical programming.

There are a number of reasons for this interest and we shall not attempt to discuss them in detail. There are, however, two general attributes which summarize many of the reasons.

One, it is a problem class that has been driven in large part by the challenge of practice. Historically, the MCF problem class was one of the first to yield applicable optimization models (e.g., the models of Hitchcock, Kantorovich, Koopmans). Today, the MCF problem

class continues to exhibit an even greater ability to model a variety of important problems. These models motivated the development of the first usable pencil and paper heuristics and optimization algorithms. In an analogous manner, new applications continue to beg the question of improved solution software.

Another attribute is its ability to indicate things to come in other fields. For instance, MCF models were not only among the first models developed but they later helped to open up the area of large-scale optimization modeling. MCF solution algorithms also pioneered the area of computer optimization software with the development of a transportation code on the National Bureau of Standards' Eastern Automatic Computer in 1952. Today MCF algorithms are continuing to lead the way in the development of more efficient computer implementation techniques for many different types of mathematical programming software.

The early algorithmic work of the 1940's and early 50's [14, 15, 42] developed heuristic and/or optimization methods which would now be called "primal simplex" methods. Following this work, a number of non-primal simplex methods were proposed including out-of-kilter, dual simplex, and others [2, 10, 32, 33].

As these later methods came along a belief developed that the primal simplex method was not an efficient solution procedure for the MCF problem class. By the mid-60's it was felt that the out-of-kilter method was the most viable approach. This belief also led to the

development of a number of specialized out-of-kilter variants for different subclasses of the MCF problem class.

It is interesting to note that this belief ignored the proposals of Glickman, L. Johnson, and Eselson [36] and E. Johnson [44] for improving the way in which the steps of the primal simplex methods are performed. Many of these proposals are embodied in contemporary implementations of the primal simplex algorithm for the MCF problem.

Contemporary MCF software development began in 1970 with the studies by Barr, Glover, and Klingman [4], Glover, Karney, and Klingman [37], Glover, Karney, Klingman, and Napier [38], Klingman, Napier, and Stutz [49], and Srinivasan and Thompson [58]. This work broke with the past in that:

1. Larger and standardized test problems were established [49].
2. Contemporary computer science methodologies were employed.
3. Primal simplex codes as well as others were developed and evaluated using the same computer and compiler.

The first comprehensive algorithm comparisons were done by [37, 38, 49] who compare out-of-kilter and primal simplex codes on a diverse set of test problems. These studies showed that the primal simplex method is substantially superior to the out-of-kilter method. The success of the primal simplex method was later independently verified by experiments [1, 3, 8, 9, 40, 48, 51, 53, 54, 55].

In addition to verifying the efficiency of the primal simplex method, these later studies proposed alternative pivot strategies and data structures that were embodied in new codes.

Two of these codes, GNET [9] and RNET]40], have received considerable attention due to their reported computational enhancements. For instance, these studies [9, 40] without comparing directly on the same computer and compiler, but running on the same problems, each claim ([9, p. 22-23] and [40, p. 18]) that their respective codes are substantially more efficient than other implementations of the primal simplex method.

These claims stimulated us to conduct a test comparing these codes to previously published results. Thus, in June 1979 we tested PNET-I [37], ARC-II [8], GNET [9], and RNET (version 3.4 [40] on the CDC 6600 at the University of Texas using the FORTRAN MNF compiler and the same test problems [8, 9, 37, 40]. (See Table VIII for NETGEN problem specifications.)

The results of this test, which are contained in Table IX, indicate that PNET-I and GNET are comparable in solution speed but inferior to ARC-II and RNET and that ARC-II and RNET are comparable in solution speed. PNET-I requires the least computer memory following by ARC-II, GNET, and RNET, respectively.

The most important point to be gleaned from this study is that computer efficiency conversion factors are not sufficiently accurate and should not be used to make code comparisons. The studies [9, 40], each of whom claimed computations superior, made this claim on the basis of computer efficiency conversion factors.

In May 1980, Glover, Klingman, Mead, and Mote initiated a study to integrate and extend the collective ideas of the 70's on MCF software development. This study is not completed due to the massive number of

TABLE VIII

PROBLEM SPECIFICATIONS

	Number of Nodes	Number of Sources	Number of Sinks	Number of Arcs	Cost Range Min	Cost Range Max	Total Supply	Transshipments Sources	Transshipments Sinks	Percent of High Cost	Percent of Arcs Capacitated	Upper Bound Range Min	Upper Bound Range Max	Random No. Seed
1.	200	100	100	1300	1	100	100,000	0	0	0	-	0	0	13502460
2.	200	100	100	1500	1	100	100,000	0	0	0	0	0	0	13502460
3.	200	100	100	2000	1	100	100,000	0	0	0	0	0	0	13502460
4.	200	100	100	2200	1	100	100,000	0	0	0	0	0	0	13502460
5.	200	100	100	2900	1	100	100,000	0	0	0	0	0	0	13502460
6.	300	150	150	3150	1	100	150,000	0	0	0	0	0	0	13502460
7.	300	150	150	4500	1	100	150,000	0	0	0	0	0	0	13502460
8.	300	150	150	5155	1	100	150,000	0	0	0	0	0	0	13502460
9.	300	150	150	6075	1	100	150,000	0	0	0	0	0	0	13502460
10.	300	150	150	6300	1	100	150,000	0	0	0	0	0	0	13502460
11.	400	200	200	1500	1	100	200	0	0	0	0	0	0	13502460
12.	400	200	200	2250	1	100	200	0	0	0	0	0	0	13502460
13.	400	200	200	3000	1	100	200	0	0	0	0	0	0	13502460
14.	400	200	200	3750	1	100	200	0	0	0	0	0	0	13502460
15.	400	200	200	4500	1	100	200	0	0	0	0	0	0	13502460
16.	400	8	60	1306	1	100	400,000	0	0	30.	20.	16,000	30,000	13502460
17.	400	8	60	2443	1	100	400,000	0	0	30.	20.	16,000	30,000	13502460
18.	400	8	60	1306	1	100	400,000	0	0	30.	20.	20,000	120,000	13502460
19.	400	8	60	2443	1	100	400,000	0	0	30.	20.	20,000	120,000	13502460
20.	400	8	60	1416	1	100	400,000	5	50	30.	40.	16,000	30,000	13502460
21.	400	8	60	2836	1	100	400,000	5	50	30.	40.	16,000	30,000	13502460
22.	400	8	60	1416	1	100	400,000	5	50	30.	40.	20,000	120,000	13502460
23.	400	8	60	2836	1	100	400,000	5	50	30.	40.	20,000	120,000	13502460
24.	400	4	12	1382	1	100	400,000	0	0	30.	80.	16,000	30,000	13502460
25.	400	4	12	2676	1	100	400,000	0	0	30.	80.	16,000	30,000	13502460
26.	400	4	12	1382	1	100	400,000	0	0	30.	80.	20,000	120,000	13502460
27.	400	4	12	2676	1	100	400,000	0	0	30.	80.	20,000	120,000	13502460
28.	1000	50	50	2900	1	100	1,000,000	0	0	0	0	0	0	13502460
29.	1000	50	50	3400	1	100	1,000,000	0	0	0	0	0	0	13502460
30.	1000	50	50	4400	1	100	1,000,000	0	0	0	0	0	0	13502460
31.	1000	50	50	4800	1	100	1,000,000	0	0	0	0	0	0	13502460
32.	1500	75	75	4342	1	100	1,500,000	0	0	0	0	0	0	13502460
33.	1500	75	75	4385	1	100	1,500,000	0	0	0	0	0	0	13502460
34.	1500	75	75	5107	1	100	1,500,000	0	0	0	0	0	0	13502460
35.	1500	75	75	5730	1	100	1,500,000	0	0	0	0	0	0	13502460

TABLE IX

SOLUTION TIMES IN SECONDS ON A
CDC 6600 USING MNF FORTRAN COMPILER

PROBLEMS	ARC-II	GNET	PNET-I	RNET (version 3.4)
1	.78	1.47	1.07	1.09
2	.89	1.48	1.25	1.07
3	1.01	2.06	1.64	1.11
4	.95	1.88	1.27	1.19
5	1.25	2.09	1.63	1.34
6	2.11	3.27	2.86	2.11
7	2.23	4.16	3.37	2.46
8	2.99	5.08	4.10	3.27
9	2.99	5.15	4.15	3.51
10	4.02	6.00	5.27	4.43
11	1.92	2.23	2.31	1.91
12	2.36	2.97	3.71	2.46
13	3.13	4.07	3.47	3.26
14	2.96	4.62	3.44	3.33
15	3.12	5.47	4.79	3.62
16	1.38	1.90	2.15	1.75
17	1.87	2.32	2.60	1.69
18	1.26	1.86	1.70	1.72
19	1.72	2.31	2.40	1.78
20	1.28	1.79	2.47	1.71
21	1.83	2.51	2.46	1.97
22	1.26	1.86	2.01	1.58
23	1.67	2.26	2.74	1.68
24	1.52	2.48	2.91	1.61
25	1.83	2.80	3.96	2.11
26	1.08	1.88	4.05	1.41
27	1.62	2.43	4.21	1.68
28	4.40	5.93	5.37	3.72
29	4.87	6.26	6.25	4.15
30	4.88	6.85	7.90	4.68
31	5.68	7.23	7.58	4.92
32	7.42	11.47	11.73	6.29
33	7.82	12.61	15.95	6.57
34	8.21	12.12	13.76	7.56
35	8.81	13.46	15.87	8.11
TOTAL	103.12	154.33	162.40	102.85

start, pivot, and data structure options to be investigated.

Preliminary results on a new code, called ARCNET, are very encouraging, however. Table X indicates a comparison of PNET-I, GNET, RNET (version 3.6) (using different pivot strategies), and ARCNET on the widely used NETGEN [49] problems of the 70's. (See Table VIII.) The times reported for ARCNET are based on using the same start and pivot criteria for all problems. Moreover, due to human time limitations this is the only start and pivot strategy which has been tested in ARCNET. The heading for RNET indicates the value used for the frequency (FRQ) parameter. (That is, RNET 5 indicates that the value of the FRQ was set to 5.)

The data in Table X have been divided into the four parts, problems 1-10 which are transportation problems, problems 11-15 which are assignment problems, problems 16-27 which are capacitated transshipment problems, and problems 28-35 which are large transportation and transshipment problems. Subtotals are provided for each problem class.

There are a number of interesting observations concerning these subtotals:

1. On transportation problems (problems 1-10) ARCNET is slightly faster than the best RNET times. All codes are substantially faster than GNET. Although ARCNET is the fastest code on these problems, it makes substantially more pivots than PNET-I and GNET.

2. The results on assignment problems (problems 11-15) are similar to those for transportation problems. ARCNET gains in superiority over the other codes and again GNET is substantially slower than all

TABLE X

SOLUTION TIMES IN SECONDS AND NUMBER OF PIVOTS ON
DUAL CYBER 170/175 USING FRN FORTRAN COMPILER

CODE:	PNET-I		GNET		ARCNET		RNET 10		RNET 5		RNET 2		RNET 1		RNET 0.5	
Problem No.	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots	Sol. Time	No. of Pivots
1	.30	374	.42	339	.25	526	.48	896	.38	649	.39	540	.36	396	.47	362
2	.31	364	.45	342	.26	529	.65	1126	.43	752	.38	501	.41	424	.48	365
3	.41	368	.60	378	.40	735	.78	1345	.59	964	.46	624	.46	501	.58	421
4	.51	502	.67	392	.46	774	.81	1343	.64	1017	.47	612	.51	524	.67	466
5	.52	453	.79	413	.50	832	.82	1445	.69	1059	.57	731	.54	541	.77	540
6	.85	792	1.10	555	.78	1204	1.42	2107	1.11	1559	.78	961	.86	850	.99	695
7	1.18	887	1.54	631	.98	1457	1.85	2770	1.46	1974	1.07	1218	.97	936	1.07	730
8	1.35	827	1.56	590	1.09	1472	2.11	3076	1.54	2022	1.10	1250	1.15	1030	1.33	861
9	1.10	717	1.72	615	1.21	1668	2.15	3192	1.81	2328	1.55	1635	1.10	1089	1.42	910
10	1.49	819	2.16	689	1.31	1782	2.23	3340	1.68	2214	1.53	1628	1.41	1257	1.52	1006
SUBTOTAL	8.02	6103	11.01	4944	7.24	10979	13.30	20640	10.33	14536	8.30	9700	7.87	7548	9.30	6356
11	.75	1594	.92	800	.37	1428	.55	1792	.67	1861	.42	868	.56	782	.85	745
12	.79	1582	1.05	838	.54	1835	.97	2966	.93	2445	.71	1406	.70	996	.93	778
13	1.09	1967	1.27	916	.75	2455	1.32	4264	1.01	2890	.86	1711	.79	1021	1.21	1007
14	.97	1490	1.44	898	.94	3127	1.51	4805	1.29	3251	1.13	2180	1.15	1551	1.24	1038
15	1.20	1784	1.85	1048	1.06	3424	1.55	4711	1.68	4385	1.21	2248	1.05	1319	1.42	1165
SUBTOTAL	4.80	8417	6.53	4500	3.66	12269	5.90	18538	5.58	14838	4.33	8413	4.25	5669	5.65	4733
16	.55	1173	.70	671	.29	803	.43	990	.46	950	.49	795	.56	640	.73	580
17	.78	1550	.83	695	.29	839	.61	1488	.46	1079	.45	727	.56	648	.76	606
18	.45	931	.67	692	.26	765	.43	925	.45	946	.48	782	.56	637	.73	584
19	.64	1342	.78	698	.25	791	.53	1301	.51	1161	.50	789	.65	723	.78	584
20	.53	1099	.64	655	.29	708	.43	930	.43	906	.50	737	.59	634	.85	585
21	.94	1702	.79	621	.30	987	.59	1507	.61	1379	.65	1062	.60	722	.78	593
22	.54	1173	.64	643	.29	736	.42	908	.41	798	.46	661	.59	619	.78	569
23	.70	1322	.75	618	.35	984	.46	1275	.47	1072	.47	803	.52	618	.71	535
24	.45	1227	.62	665	.23	833	.50	1329	.49	1080	.46	800	.62	770	.87	704
25	.84	2048	.83	732	.49	1653	.62	1859	.66	1639	.74	1406	.87	1101	1.27	952
26	.39	1103	.59	599	.20	720	.36	926	.33	729	.41	676	.46	549	.72	553
27	.73	1747	.92	829	.38	1330	.45	1400	.50	1289	.50	931	.65	825	.87	690
SUBTOTAL	7.54	16417	8.76	8118	3.62	11148	5.83	14838	5.78	13028	6.11	10169	7.23	8486	9.85	7541
28	1.62	3039	2.09	1348	.73	1878	1.03	2410	.91	1935	1.04	1566	1.26	1453	1.76	1385
29	1.56	2927	2.40	1556	.81	2402	1.08	2611	1.12	2377	1.16	1901	1.48	1748	1.93	1488
30	2.11	4108	2.89	1842	.85	2578	1.53	3862	1.24	2609	1.16	1933	1.48	1823	2.02	1592
31	1.93	3929	2.67	1682	.84	2685	1.68	4083	1.43	3051	1.29	2100	1.38	1643	1.98	1576
32	3.04	4948	4.67	2343	1.40	3624	1.75	3707	1.89	3608	1.65	2769	2.36	2629	3.02	2327
33	2.88	4603	4.49	2344	1.48	3876	1.72	3697	1.66	3229	1.99	2928	2.11	2387	2.79	2158
34	3.07	5141	4.63	2259	1.65	4100	2.01	4539	2.03	4031	1.76	2717	2.03	2402	3.06	2306
35	3.60	6031	4.88	2484	1.73	4332	2.21	4859	2.11	4280	2.12	3353	2.23	2690	3.28	2570
SUBTOTAL	19.81	34726	28.72	25858	9.49	25475	13.01	29768	12.39	25120	12.17	19267	14.33	16770	19.84	15402
GRAND TOTAL	40.17	65663	55.02	32948	24.01	59871	38.04	83784	34.08	77326	30.91	47545	33.68	38473	44.64	34032

other codes. The number of pivots made by ARCNET as compared to the other codes has increased percentagewise. (In order to gain some idea of the advantage of developing specialized algorithms for the assignment problem, the reader may find it interesting to compare the times in Tables II and X. The assignment problems, computer, and compiler are the same.)

3. On the capacitated transshipment problems (problems 16-27), ARCNET performs extremely well as compared to the other codes being approximately 60% faster than the closest competitor, RNET 5.

4. On the large problems (problems 28-35), ARCNET is again faster being approximately 30% faster than the next closest times of RNET 2.

The total solution times indicate that ARCNET is the fastest followed by RNET 2. This is somewhat surprising given that the start and pivot strategies of ARCNET have not been tuned. Another surprising result is that PNET-I is much faster than GNET on the Dual CYBER 170/175 using the FTN FORTRAN compiler since on the CDC 6600 using the MNF FORTRAN compiler (see Table IX) they are approximately equal. This illustrates again the importance of conducting tests on the same computer and compiler.

REFERENCES

1. R. Armstrong, D. Klingman, and D. Whitman, "Implementation and Analysis of a Variant of the Dual Method for the Capacitated Transshipment Problem." Research Report CCS 324, Center for Cybernetic Studies, The University of Texas at Austin. To appear in *European Journal of Operations Research*, 4, 6.

2. E. Balas and P. L. Hammer, "On the Transportation Problem--Part I." *Cahiers du Centre d'Etudes de Recherche Operationelle*, 4, 2 (1962) 98.

3. R. Barr, J. Elam, F. Glover, and D. Klingman, "A Network Augmenting Path Basis Algorithm for Transshipment Problems." *Lecture Notes in Economics and Mathematical Systems - 174, Extremal Methods and Systems Analysis*, A. Fiacco and K. Kortanek, eds., Springer-Verlag, Berlin, 1980.

4. R. Barr, F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes." *Mathematical Programming*, 7, 1 (1974) 60-87.

5. R. Barr, F. Glover, and D. Klingman, "The Alternating Basis Algorithm for Assignment Problems." *Mathematical Programming*, 13 (1977) 1-13.

6. R. Barr, F. Glover, and D. Klingman, "A New Alternating Basis Algorithm for Semi-Assignment Networks." *Proceedings of the Bicentennial Conference on Mathematical Programming*, Gaithersburg, Maryland, 1977.

7. R. Barr, F. Glover, and D. Klingman, "The Generalized Alternating Path Algorithm for Transportation Problems." *European Journal of Operations Research*, 2 (1978) 137-144.

8. R. Barr, F. Glover, and D. Klingman, "Enhancements of Spanning Tree Labeling Procedures for Network Optimization," Research Report CCS 262, Center for Cybernetic Studies, The University of Texas at Austin, 1976. *INFOR*, 17, 1 (1979) 16-34.

9. G. Bradley, G. Brown, and G. Graves, "Design and Implementation of Large-Scale Primal Transshipment Algorithms." *Management Science*, 24, 1 (1977) 1-34.

10. R. Busacker and P. Gowen, "A Procedure for Determining a Family of Minimum-Cost Network Flow Patterns." ORO Technical Report 15, Operations Research Office, Johns Hopkins University, 1961.

11. T. Cheung, "Computational Comparison of Eight Methods for the Maximum Network Flow Problems." Technical Report 78-07, Department of Computer Sciences, University of Ottawa, Ontario, 1978.

12. W. H. Cunningham, "A Network Simplex Method." *Mathematical Programming*, 11 (1976) 105-116.

13. W. H. Cunningham, "Theoretical Properties of the Network Simplex Method." *Mathematics of Operations Research*, 4 (1979) 196-208.

14. G. Dantzig, "Application of the Simplex Method to a Transportation Problem." in *Activity Analysis of Production and Allocation*, T. C. Koopmans, ed., John Wiley and Sons, New York, 1951.

15. G. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey, 1963.

16. G. Dantzig and D. Fulkerson, "On the Max-Flow Min-Cut Theorem of Networks." *Annals of Mathematical Studies*, Princeton University Press, Princeton, N.J. (1956) 215-221.

17. E. Denardo and B. Fox, "Shortest-Route Methods: 1. Reaching, Pruning, and Buckets." *Operations Research*, 27, 1 (1979) 161-186.

18. R. Dial, "Algorithm 360 Shortest Path Forest with Topological Ordering." *Communications of the ACM*, 12 (1969) 632-633.

19. R. Dial, F. Glover, D. Karney, and D. Klingman, "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees." *Networks*, 9 (1979) 215-248.

20. E. Dijkstra, "A Note on Two Problems in Connexion with Graphs." *Numerical Mathematics*, 1 (1959) 269-271.

21. E. Dinic, "Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation." *Soviet Math. Doklady*, 11 (1970) 1277-1280.

22. E. Dinic and M. Kronrod, "An Algorithm for the Solution of the Assignment Problem." *Soviet Math. Doklady*, 10, 6 (1969).

23. S. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms." *Operations Research*, 17 (1969) 395-412.

24. J. Edmonds and R. Karp, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems." *Journal of the Association for Computing Machinery*, 19 (1972) 248-264.

25. J. Elam, D. Klingman, and J. Mulvey, "An Evaluation of Mathematical Programming and Minicomputers." *European Journal of Operations Research*, 3, 1 (1979) 30-39.

26. S. Elmaghraby and J. Moder, eds., *Handbook of Operations Research: Foundations and Fundamentals*, Van Nostrand Reinhold Company, New York, 1978.

27. M. Engquist, "A Successive Shortest Path Algorithm for the Assignment Problem." To appear as a research report, Center for Cybernetic Studies, The University of Texas at Austin.

28. M. Florian, S. Nguyen, and S. Pallottino, "Dual Simplex Approach to Finding All Shortest Paths." Publication #39, Transportation Research Center, University of Montreal, 1979. To appear in *Networks*.

29. C. Fong and M. Rao, "Accelerated Labeling Algorithms for the Maximal Flow Problem with Applications to Transportation and Assignment Problems." Working Paper 7222, Graduate School of Business, University of Rochester, 1974.

30. L. Ford and D. Fulkerson, "Maximal Flow Through a Network." *Canadian Journal of Mathematics*, 8 (1956) 399-404.

31. L. Ford and D. Fulkerson, "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem." *Naval Research Logsitics Quarterly*, 4, 1 (1957) 47.

32. L. Ford and D. Fulkerson, *Flows in Networks*. Princeton University Press, Princeton, N.J., 1962.

33. D. Fulkerson, "An Out-of-Kilter Method for Minimal-Cost Flow Problems." *SIAM Journal of Applied Mathematics*, 9, 1 (1961) 18.

34. D. Fulkerson and G. Dantzig, "Computations of Maximal Flows in Networks." *Naval Research Logistics Quarterly*, 2 (1955) 277-283.

35. J. Gilsinn and C. Witzgall, "A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees." NBS Technical Note 772, U.S. Department of Commerce, 1973.

36. S. Glickman, J. Johnson, and L. Eselson, "Coding the Transportation Problem." *Naval Research Logistics Quarterly*, 7, 2 (1960) 169.

37. F. Glover, D. Karney, and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems." *Networks*, 4, 3 (1974) 191-212.

38. F. Glover, D. Karney, D. Klingman, and A. Napier, "A Comparison of Computational Times for Various Starting Procedures, Basis Change Criteria and Solution Algorithms for Transportation Problems." *Management Science*, 20, 5 (1974) 793-814.

39. F. Glover, D. Klingman, J. Mote, and D. Whitman, "Comprehensive Computer Evaluation and Enhancement of Maximum Flow Algorithms." Research Report CCS 356, Center for Cybernetic Studies, The University of Texas at Austin, 1979.

40. M. Grigoriadis and T. Hsu, "The Rutgers Minimum Cost Network Flow Subroutine." *Sigmap*, 26 (1979) 17-18.

41. R. Hatch, "Bench Marks Comparing Transportation Codes based on Primal Simplex and Primal-Dual Algorithms." *Operations Research*, 23, 6 (1975) 1167.

42. F. Hitchcock, "The Distribution of a Product from Several Sources to Numerous Localities." *Journal of Mathematics and Physics*, 20, 2 (1941) 224.

43. M. Hung and W. Rom, "Solving the Assignment Problem by Relaxation." To appear in *Operations Research*.

44. E. Johnson, "Networks and Basic Solutions." *Operations Research*, 14 (1966) 619-623.

45. D. Karney and D. Klingman, "Implementation and Computational Study on an In-Core Out-of-Core Primal Network Code." *Operations Research*, 24, 6 (1976) 1056-1077.

46. D. Klingman, J. Mote, and D. Whitman, "Computational Analysis of In-Core Out-of-Core Shortest Path Algorithms." Research Report CCS 322, Center for Cybernetic Studies, The University of Texas at Austin, 1978.

47. D. Klingman and J. Mulvey, "Applicability of Shortest-Path and Minimum-Cost Flow Network Algorithms for Minicomputers and Micro-Processors." *Proceedings of the 9th International Mathematical Programming Symposium*, Budapest, August 23-27, 1976.

48. D. Klingman, A. Napier, and T. Ross, "A Computational Study of the Effects of Problem Dimensions on Solution Times for Transportation Problems." *Journal of the Association for Computing Machinery*, 22, 3 (1975) 413-424.

49. D. Klingman, A. Napier, and J. Stutz, "NETGEN--A Program for Generating Large-Scale (Un)Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems." *Management Science*, 20, 5 (1974) 814-821.

50. M. Kuhn, "A Primal Method for Minimal Cost Flows with Application to the Assignment and Transportation Problems." *Management Science*, 14, 3 (1967) 205.

51. R. Langley, J. Kennington, and C. Shetty, "Efficient Computational Devices for the Capacitated Transportation Problem." *Naval Research Logistics Quarterly*, 21, 4 (1974) 637.

52. V. Malhutra, M. Kumar, and S. Maheshwari, "An $O(|V|^3)$ Algorithm for Finding Maximum Flows in Networks." *Information Processing Letters*, 7, 6 (1978) 277-278.

53. R. McBride, "Factorization in Large-Scale Linear Programming." Working Paper No. 200, Western Management Science Institute, UCLA, 1973.

54. J. Mulvey, "Testing of a Large-Scale Network Optimization Program." *Mathematical Programming*, 15 (1978) 291-314.

55. J. Mulvey, "Pivot Strategies for Primal-Simplex Network Codes." *Journal of the Association for Computing Machinery*, 25, 2 (1978) 266-270.

56. U. Pape, "Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem." *Mathematical Programming*, 7 (1974) 212-222.

57. S. Phillips and M. Dessouky, "The Cut Search Algorithm with Arc Capacities and Lower Bounds." *Management Science*, 25 (1979) 396-404.

58. V. Srinivasan and G. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm." *Journal of the Association for Computing Machinery*, 20, 2 (1973) 194.

59.

60. A. Weintraub and F. Barahona, "A Dual Algorithm for the Assignment Problem." Publication No. 79/02/C, Departamento de Industrias, Universidad de Chile-Sede Occidente, Santiago, Chile, 1979.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Center for Cybernetic Studies The University of Texas at Austin	Unclassified
	2b. GROUP

3. REPORT TITLE

Recent Developments in Computer Implementation Technology for Network Flow Algorithms

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

Fred Glover and Darwin Klingman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
July 1980	33	60

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-80-C-0242 and N00014-78-C-0222	CCS-377
b. PROJECT NO. NR047-071	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Office of Naval Research (Code 434) Washington, DC

13. ABSTRACT

The application of computer implementation technology to network optimization has brought about unprecedented advances in solution efficiency. The remarkable gains of the early to mid 1970's for solving transportation and transhipment problems are widely known, enabling network codes to out-perform LP codes by two orders of magnitude for these problems. The pioneering study by Gilsinn and Witzgall demonstrated that effective use of computer implementation technology could reduce solution times for shortest path problems from one minute to slightly more than one second, using the same general shortest path algorithm, computer, and compiler.

The momentum launched by these studies has not dwindled, but continues into the present. New advances in all areas of network optimization have recently superseded the procedures previously found to be best. Latest computer implementations clearly outstrip the best codes of the recent past as our understanding of the important relation between algorithmic design and implementation continues to grow.

We undertake to report on some of the major computer implementation studies of the past few years and to present preliminary results on the new developments.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

A-31408

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Shortest Path						
Assignment Problems						
Transportation Problems						
Network Flow						
Transhipment Problems						